LJ-88-28

QL
155
.L6

NATIONAL MARINE FISHERIES SERVICE
SOUTHWEST FISHERIES CENTER
P.O. BOX 271
LA JOLLA, CA 92038

LIBRARY
DEC 6 1988
National Marine Fisheries Service
2570 Dole Street
Honolulu, Hawaii

SOUTHWEST FISHERIES CENTER

NOVEMBER 1988

A MODEL OF TUNA VESSEL AND DOLPHIN SCHOOL MOVEMENT IN THE EASTERN TROPICAL PACIFIC OCEAN: TECHNICAL DESCRIPTION OF MODEL

by

Pierre Kleiber and Elizabeth F. Edwards

ADMINISTRATIVE REPORT LJ-88-28

This Administrative Report is issued as an informal document to ensure prompt dissemination of preliminary results, interim reports and special studies. We recommend that it not be abstracted or cited.

A Model of Tuna Vessel and Dolphin School Movement in the Eastern Tropical Pacific Ocean: Technical Description of Model

Pierre Kleiber and Elizabeth F. Edwards

Southwest Fisheries Center
National Marine Fisheries Service
La Jolla, California

November 1988

ADMINISTRATIVE REPORT LJ-88-28

TABLE OF CONTENTS

1. Introduction 1

2. General Approach 2

3. Specifics 4

3.1. Procedure for Moving Objects in One Time Step . . 4

3.2 Environmental Quality Topography for Dolphins . . 7

3.3. Dolphin Movement Rules 8

3.4 Vessel Movement Rules 10

3.5 School Sightings 13

3.5.1 Determining when a sighting occurs 13

3.5.2 What happens when a sighting occurs 15

4. Initial Results 16

LIST OF BOXES

Box 1. Overview of model logic 3

Box 2. Algorithm to check if vessel has encountered a school 15

LIST OF FIGURES

Figure 1. Two segments of a hypothetical course track . . . 4

Figure 2. The Circular-Normal distribution 6

Figure 3. Two examples of environmental quality topography . 9

Figure 4. Dolphin k vs. environmental quality gradient . . . 10

Figure 5. Dolphin speed vs. environmental quality 11

Figure 6. Example time series of success variable 12

Figure 7. Vessel k vs. success variable 12

Figure 8. Search area for a vessel in one time step 14

Figure 9. Vessel course tracks during 200 time steps 17

LIST OF APPENDICES

Appendix A. FORTRAN code for TOPS model program A1

Appendix B. Program for Circular-Normal distribution B1

A Model of Tuna Vessel and Dolphin School Movement in the Eastern Tropical Pacific Ocean: Technical Description of Model

Pierre Kleiber and Elizabeth F. Edwards

1. Introduction

Observers on tuna purse-seine vessels fishing in the eastern tropical Pacific Ocean (ETP) record the time of sighting and position of dolphin schools encountered during the searching process for tuna. Dolphin schools are objects of search by these tuna vessels because the schools are often associated with the tuna these fishermen seek. The United States National Marine Fisheries Service would like to use these tuna vessel observer data (TVOD) to monitor trends in abundance of the dolphins. Suspected non-randomness in search procedures and in dolphin school distributions violates the basic assumptions of most methods currently available for trend monitoring. This is true in particular for the most popular method, line transect analysis (e.g. Burnham et al. 1980, Hammond and Laake 1983, Buckland and Anganuzzi 1988).

To investigate quantitatively and qualitatively the effect of various types of non-randomness on estimates derived from sightings data, we have developed a computer model to simulate the TVOD collection process (TOPS: Tuna-vessel Observer Program Simulator). This report describes the technical aspects of TOPS development and structure. Results from initial simulations appear separately (Edwards and Kleiber in prep.).

In general, TOPS is a spatial model in which 1) dolphin schools move in a non-determinate but also non-random fashion so that they tend to form clumps of higher than average density, in response to a non-random topography of "environmental quality", 2) tuna vessels move non-randomly in response to encounters with dolphin schools, and 3) TVOD collected by "observers" on the

simulated vessels is directed to an output file listing the time and position of each dolphin school encountered.

2. General Approach

TOPS tracks exact positions of individual objects (dolphin schools and tuna vessels) rather than charting the number of objects in, and exchanges between, discrete geographic grid squares. This exact tracking formulation simulates more closely the actual process of recording school sightings, which are individual events involving a single school, a single vessel, and a precise position.

Spatially, TOPS simulates a square of open ocean 1200 nautical miles (nmi) on a side. The model can accommodate up to 100 tuna vessels and 3000 dolphin schools during any single simulation. In this initial stage of development all vessels have identical rules of behavior and do not interact with each other. The same is true for dolphin schools, although the movement rules are different for vessels and schools. Interaction between vessels and schools is, of course, included in the model.

Non-random distributions of dolphin schools in the model result from school reactions to the local level and gradient of an underlying "environmental quality topography". Dolphin schools aggregate in areas of high "quality" and avoid areas of low "quality".

Non-random distributions of tuna vessels result from vessel responses to dolphin school sightings. Movement rules for vessels are designed so that vessels behave differently in dolphin-rich as

opposed to dolphin-poor areas. Vessels spend a disproportionate amount of time in dolphin-rich areas.

Simulation progresses in discrete time steps of one hour. A schema of the model logic is given in Box 1. Details of the various model operations are given in the next section. A full listing of the TOPS model program in FORTRAN is given in Appendix A.

Box 1. Overview of model logic.

```
Set-up
   read input data
   assign initial positions of schools and dolphins

Loop through time steps

   Loop through schools
      determine new direction
      determine speed
      update school position
   End school loop

   Loop through vessels
      if school is sighted in this time step then
         move vessel to school position, put in fish handling
            mode, and increment success variable
         move school elsewhere and put in flight (fright) mode
      otherwise
         choose new direction
         update vessel position
      end-if
      output vessel position and whether school encountered
   End vessel loop

End time step loop
```

3. Specifics

3.1. Procedure for Moving Objects in One Time Step

To calculate a new geographic position for a dolphin school or tuna vessel, it is necessary to know the speed and direction of movement. The position of each object at time t is specified by a vector, $[x_t,y_t]$, and direction by a unit vector, $[dx_t,dy_t]$ (Figure 1). Position and direction vectors for all objects are maintained in memory in two position arrays and two direction

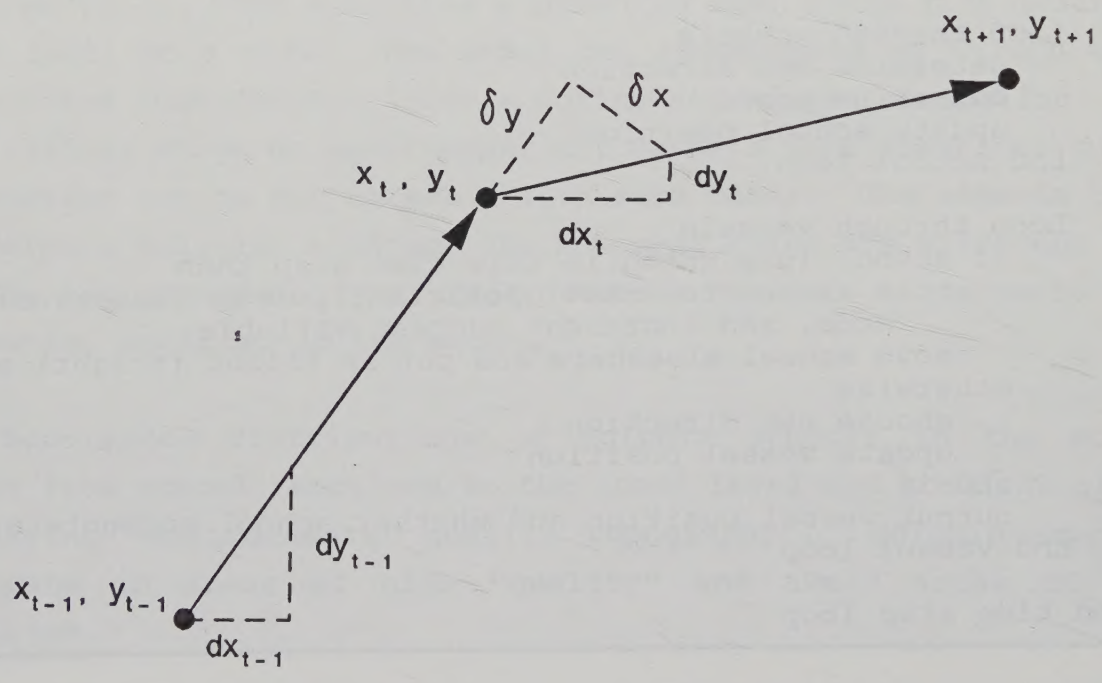


Figure 1. Two segments of a hypothetical course track showing positions at times t-1, t, and t+1. dx_t and dy_t specify the direction at time t. δx and δy specify the direction at time t+1 relative to the direction at time t.

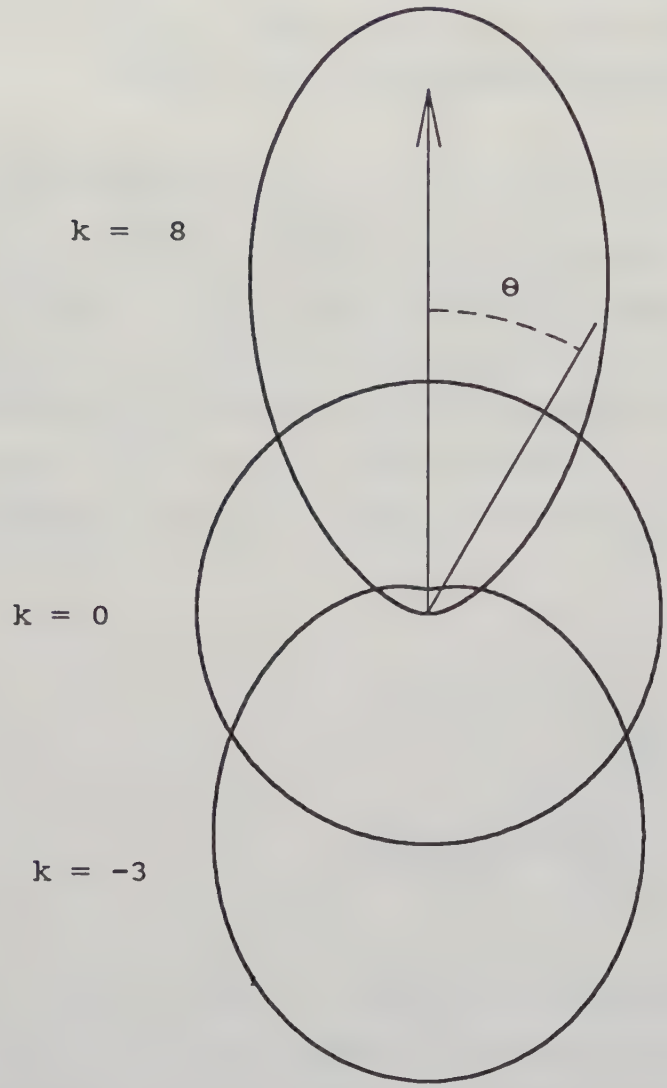
arrays, one of each type for vessels and one of each type for schools. The speed of each object is determined de novo each time step according to the existing situation of the object.

Each time step (hour), a choice of a new direction must be made for each vessel and every 10 time steps (10 hours) for each dolphin school. A choice of new direction relative to the existing direction is made stochastically by determining a probability level, P, from a uniform [0 - 1] random number generator and then calculating the associated direction, $\theta(P)$. The mapping of P to $\theta(P)$ is the inverse of the Circular-Normal probability function (Batschelet 1965). The cumulative form of this probability function is

$$(1) \qquad P(\theta) = \frac{\int_0^{\theta} e^{k\cos(\alpha)}\, d\alpha}{\int_0^{2\pi} e^{k\cos(\alpha)}\, d\alpha}$$

where the parameter, k, regulates the degree of bias in choice of a new direction, its sign determining whether the bias is in a forward or backward direction (Figure 2). The value of k for each school and vessel depends upon the circumstances of that school or vessel at that point in time.

To evaluate the inverse of Equation 1 whenever needed would be very time-consuming. Instead we prepared a table of the inverse probability function to be read into memory during set-up for a simulation (Appendix B). This table gives direction relative to existing direction for 100 even steps in probability from 0 to 1 for a series of values of the parameter k. To save more



$$p(\theta) = \frac{e^{k\cos(\theta)}}{\int_0^{2\pi} e^{k\cos(\alpha)}\,d\alpha}$$

Figure 2. The Circular-Normal distribution, which gives the probability of a new direction, θ, relative to the current direction (arrow). The square root of the probability density function is plotted in polar coordinates centered on the base of the arrow. Thus the area enclosed by any two radial lines and the plotted line is proportional to the probability that the new direction will be between the radial lines. The parameter, k, determines to what degree the directional choice is biased either in a forward direction (k positive), or in a backward direction (k negative), or whether there is an equal chance of choosing any direction (k zero).

computational time we avoided transcendental functions during the simulation by tabulating direction not as θ, but as the unit vector, $[\delta x, \delta y]=[\sin(\theta),\cos(\theta)]$. The new direction for an object is obtained from

$$(2) \qquad \begin{aligned} dx_{t+1} &= \delta y\, dx_t + \delta x\, dy_t \\ dy_{t+1} &= \delta y\, dy_t - \delta x\, dx_t \end{aligned}$$

In most cases the new position is then given by

$$(3) \qquad \begin{aligned} x_{t+1} &= (x_t + s\,\Delta t\, dx_t) \bmod x_{mx} \\ y_{t+1} &= (y_t + s\,\Delta t\, dy_t) \bmod y_{mx} \end{aligned}$$

where x_{mx} and y_{mx} are the dimensions of the simulation area (both 1200 nmi), s is the speed, and Δt is the time step. The modulus function insures that objects moving off the edge of the simulation area are reinserted on the other side, in effect giving the simulation "universe" a torroidal topology. The exception to use of Equations 3 is when a vessel encounters a school. Details are given below in section 3.5.2.

3.2 Environmental Quality Topography for Dolphins

Because the goal of initial simulations was to investigate in general the effects of clustering on various spatial scales rather that to investigate the effects of specific oceanographic parameters that might cause dolphins to aggregate, TOPS simply generates an arbitrary environmental quality topography consisting of peaks generated by the following function:

$$(4) \qquad q(x,y) = \left(\left[1 + \sin\left(\frac{2\pi n x}{x_{mx}} - \frac{\pi}{2}\right)\right] \left[1 + \sin\left(\frac{2\pi n y}{y_{mx}} - \frac{\pi}{2}\right)\right] \right)^{7}$$

where the quality, q(x,y), varies between 0 and 1, n^2 is the number of peaks in the simulated region, and γ is a parameter which adjusts the shape (peakedness) of the peaks (Figure 3). To avoid time-consuming computation of the above function during simulation, a 120 by 120 matrix of values is calculated during the set-up phase. The matrix corresponds to a square grid of geographic points 10 nmi apart. The model dolphins therefore actually perceive the quality at the nearest 10 nmi grid point.

The environmental quality topography can either be stationary or can move during simulation, much the way oceanographic features move. This is accomplished by including an offset in calculating one of the indices of the quality matrix. If the offset is constant, the topography remains stationary. If it is made to increase with time, the topography moves. Modular arithmetic is used in incrementing the offset so that the topography "wraps" around as it spills off the edge of the model "universe".

3.3. Dolphin Movement Rules

For dolphin schools, k of the Circular-Normal distribution depends on the environmental quality gradient as seen by the particular school during the last time step. The gradient is given by

$$g = \frac{q(x_t,y_t) - q(x_{t-1},y_{t-1})}{q(x_{t-1},y_{t-1}) + \epsilon} \tag{5}$$

where ϵ is a small positive number (0.001) which prevents division by zero in the case where the quality was zero at time t-1. The relationship between k and g (Figure 4) is such that for negative gradients k is also negative, thus favoring a backward choice of direction. Therefore when things are getting worse for a school,

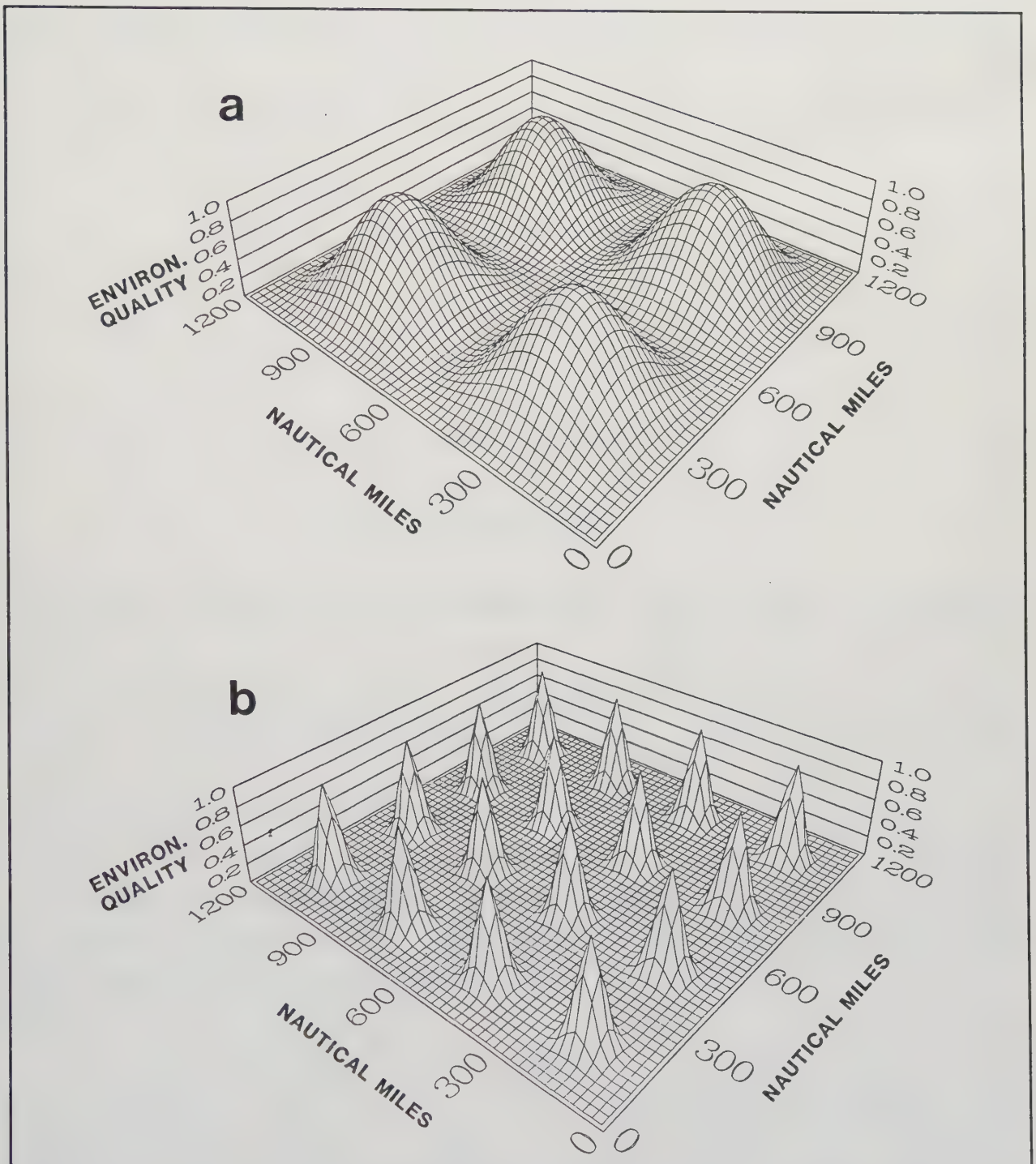


Figure 3. Two examples of environmental quality topography with different peak densities and peak shapes (a: γ=1, b: γ=5).

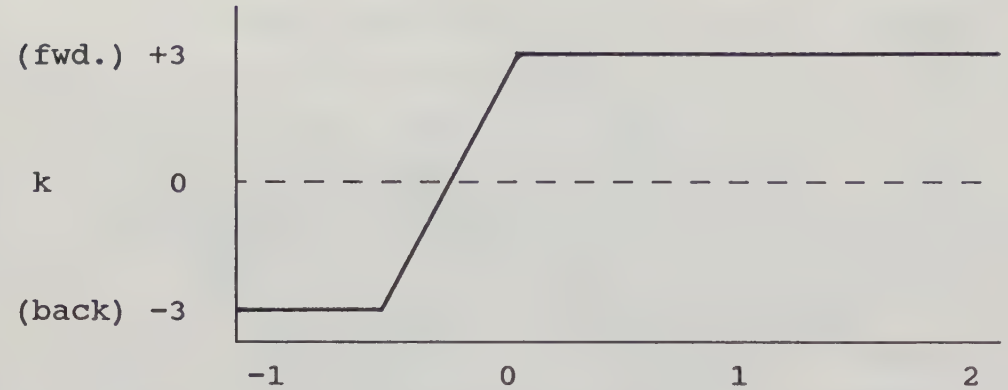


Figure 4. Dolphin k vs. environmental quality gradient along the dolphin's course track.

it tends to reverse its direction. On the other hand, if things are getting better, it tends to maintain its direction.

In addition to changing their direction, schools adjust their speed inversely with environmental quality (Figure 5). This is because in the environmental quality topography the gradients at peak tops and valley bottoms are similar (ie. close to zero). In the model dolphin schools behave differently in the two cases, moving quickly when environmental quality is bad and slowly when quality is good.

3.4 Vessel Movement Rules

For vessels, k depends on a dynamic success variable which is maintained separately for each vessel. This variable is incremented by 1 each time a vessel encounters a dolphin school,

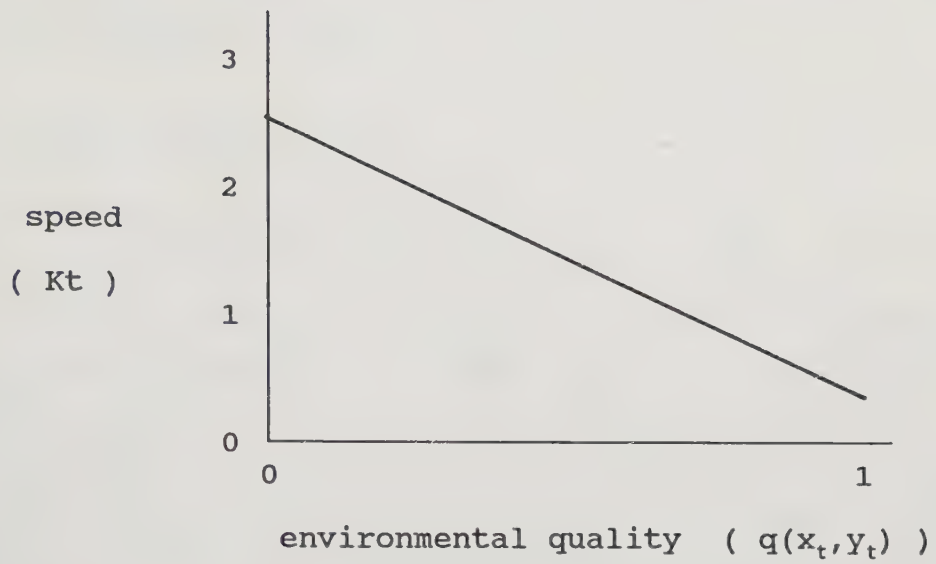


Figure 5. Dolphin speed vs. environmental quality.

but it also decays exponentially with time (Figure 6). This success variable measures the skipper's perception of dolphin density. The proportion of the success variable remaining after one time step (the McGlade coefficient) determines whether skippers are "stochasts" or "cartesians" in the terminology of McGlade and Allen (1985). Stochasts are risk-takers. When local conditions show signs of deteriorating, they are eager to move to new areas in hopes of finding better conditions. Cartesians delay moving on until local conditions have seriously deteriorated.

In the model there is an inverse relationship between k and the success variable (Figure 7). When the success variable is low (few encounters lately), then k is high, biasing the direction choice in the forward direction. In this case a vessel tends to maintain the same heading and progresses rapidly away from the region of low encounter rate. Conversely, when success is high,

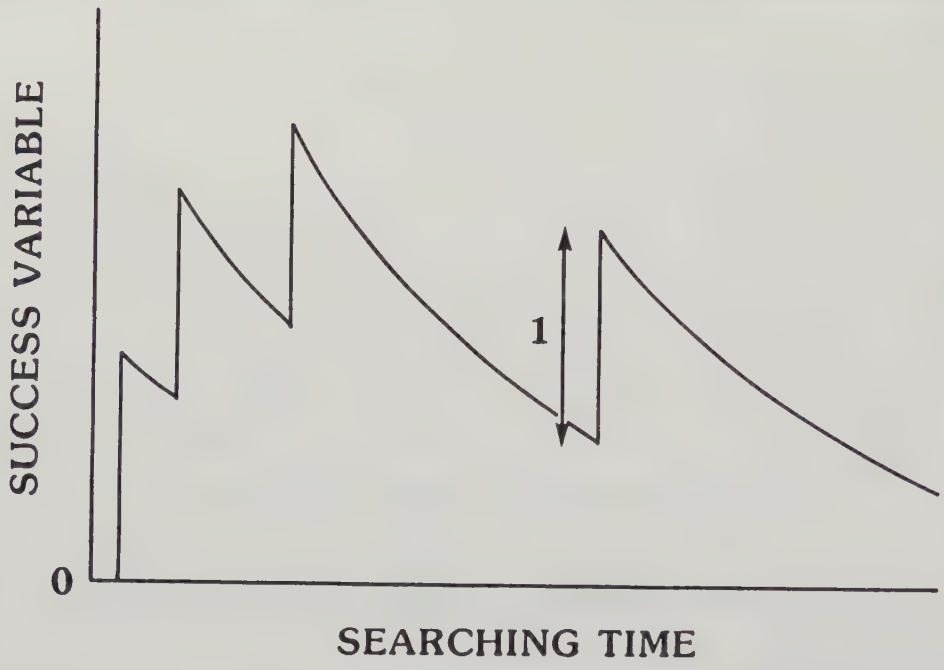


Figure 6. Example time series of success variable.

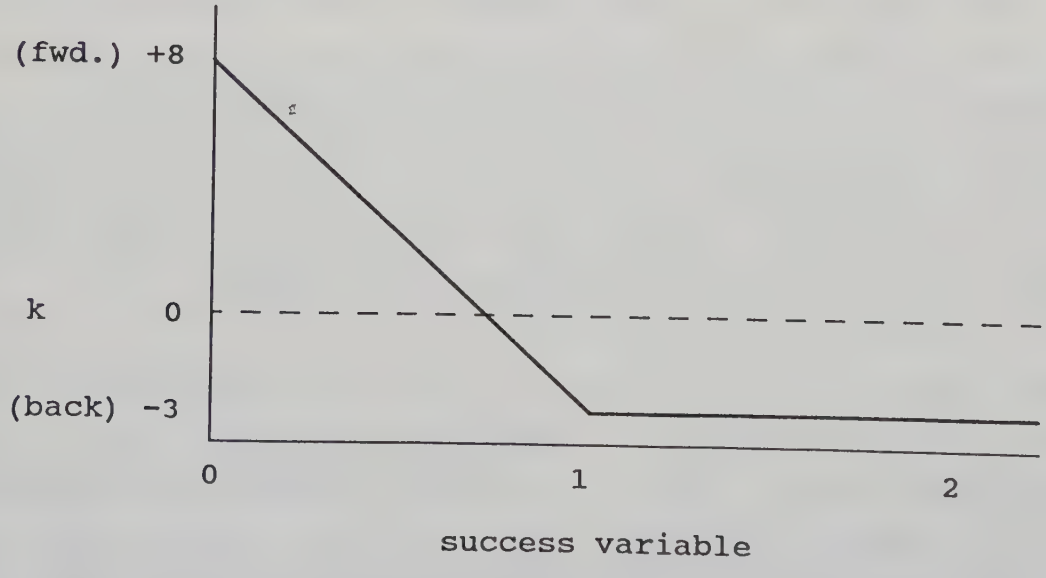


Figure 7. Vessel k vs. success variable.

the direction choice is biased in the backward direction. In this case a vessel tends to reverse its direction each time step and thereby remains in the same neighborhood.

3.5 School Sightings

3.5.1 Determining when a sighting occurs

An encounter occurs when a school comes within the search path of a vessel. For a real observer, the probability of seeing a school decreases with distance from the observer, reaching zero probability at the horizon (~7 nmi from the observing platform of a tuna vessel in the ETP). The sighting probability does not drop abruptly from 1 to 0 at a particular distance; it decreases more gradually than that. Therefore, at any particular intermediate distance, some schools that are closer than that distance are missed and some beyond it are seen. The effective sighting radius, r, is defined such that the number of schools that are within distance r and are missed is the same on average as the number that are beyond r and are seen. Thus an observer on a moving vessel in effect samples a path 2r wide. For tuna vessels, r is approximately 2 nmi.

In the model, observers are assumed to search from directly abeam to directly forward. Therefore in one time step a vessel sweeps out a rectangular search area 2r wide and $s\,\Delta t$ long (15 nmi) with an additional semicircular area of radius r (Figure 8).

For each time step and each vessel the model checks for schools within the search area using the algorithm in Box 2. Executing the encounter check algorithm for all 2500 schools for each of the 75 vessels would consume a great deal of computer time per time step. Therefore the model checks only those schools that

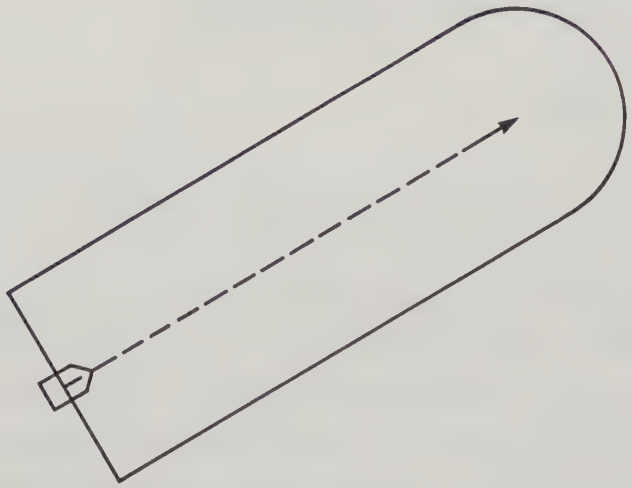


Figure 8. Search area for a vessel in one time step of the model.

are in a defined neighborhood of each vessel. This is accomplished by subdividing the simulation area into a 120 by 120 grid of geographic squares 10 nmi on a side. The model maintains a corresponding 120 by 120 matrix of entry pointers to linked lists, each list giving the index numbers of schools positioned within the associated grid square. The index number of a school in turn points to the school's geographic position in the school position array. Using the linked lists the model efficiently checks only the schools in a given grid square. Because vessels travel no more than 15 nmi in a time step, it is only necessary to check the schools in the square the vessel occupies at the start of the time step and in 2 layers of concentric neighboring squares. In this way only a fraction of the total number of schools have to be checked for each vessel.

Because schools can move from one grid square to another, there is a computational overhead with the linked lists. A school

Box 2. Algorithm to check if vessel has encountered a school.

x_b, y_b = vessel position at start of time step
dx, dy = vessel direction ; s = vessel speed
x_s, y_s = position of school
Encounter = False
$\Delta x = x_s - x_b$; $\Delta y = y_s - y_b$
$y' = \Delta x\, dx + \Delta y\, dy$
if [$y' > 0$] ! *forward of starting*
$x' = |\Delta x\, dy - \Delta y\, dx|$! *position of vessel*
if [$x' \leq r$] ! *in range of vessel track*
if [($y' \leq s\,\Delta t$) or ! *in rectangle*
($y'^2 - 2y' + x'^2 + (s\,\Delta t)^2 \leq r^2$)] ! *in semi-circle*
Encounter = True
end if
end if
end if

that migrates from one square to another must be excised from one list and added to another. Furthermore, this operation requires that we keep backward pointers as well as forward pointers for each list. But this maintenance operation requires only a few shufflings of integers in memory, thereby saving many passes through the encounter check algorithm and thus many time-consuming arithmetic operations in floating point.

3.5.2 What happens when a sighting occurs

If more than one school is in the search area of a vessel during a time step, the school closest to the vessel's position at the start of the time step is chosen. The vessel's position is advanced to the position of the school, and the vessel is put into a 5 hour waiting mode during which it does not move or sight

schools. This simulates cessation of search activity while fish are caught and loaded on board. The sighted school is moved a random distance (up to 50 nmi) away and put into an invisible mode for 5 hours, simulating fast flight and temporary breakup of a dolphin school released from a net.

To keep track of the area searched by each vessel, a TVOD record is generated for each vessel and each time step whether or not a sighting occurred. The record includes the time, vessel position, and a sighting code. The sighting code is the school index if a sighting occurred or zero if there was no sighting.

4. Initial Results

Initial simulations show that the TOPS model behaves as planned. Dolphin schools aggregate on environmental quality peaks, and vessels cluster on dolphin aggregations (Figure 9).

The primary objective for the TOPS model is to generate simulated TVOD under various conditions of non-random search, but it also can be a tool for viewing patterns of movement and aggregation. It is interesting, for example, to experiment with the effect of the McGlade coefficient on patterns of vessel movement. The printed page is a poor way to display such dynamic phenomena. We have made an animated film showing school and vessel positions for one model run, but as yet the best rendering we have of model behavior is a demonstration version running on an AT microcomputer with EGA graphics. Unfortunately, to approach the speed necessary for an animated display, this version is limited to a 300 by 300 nmi area, 6 vessels, and 200 schools, but it allows movement to be visualized and also hands-on experimentation with parameter values. This version is available from us on diskette.

Figure 9. Vessel course tracks during 200 time steps with stationary environmental topography as in Fig. 3b.

References

Batschelet, E. 1965. Statistical methods for the analysis of problems in animal orientation and certain biological rhythms. Am. Inst. Biol. Sci. Monogr. 57pp.

Buckland, S.T. and A. A. Anganuzzi. 1988. Trends in abundance of dolphins associated with tuna in the eastern tropical Pacific. Rep. Int. Whaling Comm. 38:000-000 (in press).

Burnham, K.P., Anderson, D.R., and Laake, J.L. 1980. Estimation of density from line transect sampling of biological populations. Wildl. Managr. 72:202.

Edwards, E.F. and Kleiber, P. in prep. Effects of Non-randomness on Line Transect Estimates of Dolphin School Abundance. E.F. Edwards, Southwest Fisheries Center, Box 271, La Jolla, CA 92038.

Hammond, P.H. and J.L. Laake. 1983. Trends in abundance of dolphins (Stenella spp. and Delphinus delphis) involved in the purse-seine fishery for tunas in the eastern tropical Pacific Ocean, 1977-81. Rep. Int. Whaling Comm. 34:565-583.

McGlade, J. and P.M. Allen. 1985. The fishery industry as a complex system. in: Mohn,.E.R. [ed]. Toward the inclusion of fishery interactions in management advice. Can. Tech. Rep. Fish. Aquat. Sci., No. 1347.

Appendix A. FORTRAN code for TOPS model program.

```
************************************************************************
*         Tuna-vessel Observer Program Simulator  (TOPS)
*
*  Records position, direction, and status for each school and vessel
*  each time step in file, 'plotdat'.  Also makes an observer data record
*  for each vessel and time step in file, 'obsdat'.
*
*  Kleiber,  88/3/3
*  adapted for Cray fortran, 88/4/13
************************************************************************


      CLICHE TOPSCOMM
************************************************************************
*   Define parameters and common block.
*
*  (CLICHE TOPSCOMM to ENDCLICHE is inserted in the source code wherever
*  "USE TOPSCOMM" appears.)
************************************************************************

      PARAMETER (TOTSIZ=1200)          ! simulation area dimension in n-miles
      PARAMETER (STEP=1.)              ! time step in hours
      PARAMETER (MAXLOOPS=600)         ! max # time steps

      PARAMETER (NQ=120)               ! number environmental quality grid points (in one dimension)
      PARAMETER (NPEAK=4               ! number environmental quality peaks (in one dimension)
      PARAMETER (EX=5.)                ! environmental quality peak shape
      PARAMETER (IQMOVE=10)            ! steps per environmental quality shift
      PARAMETER (KLEVELS=22)           ! max # levels of k in circ. normal formula

      PARAMETER (NSCHOOL=2500)         ! number of dolphin schools
      PARAMETER (DSPEED=2.4)           ! dolfin school speed in knots
      PARAMETER (NDSTEP=10)            ! steps per choice of dolphin direction
      PARAMETER (SPOOK=50.)            ! distance that sighted schools are "spooked" to in n-miles
      PARAMETER (QOLDMIN=.001)         ! min denominator in formula for dolphin direction choice
      PARAMETER (SGMIN=-.5, SGMAX=0.)  ! limits of gradient variable for dolphins
      PARAMETER (SKMIN=-3., SKMAX=3.)  ! associated limits of k
      PARAMETER (NGRID=120)            ! number school pointer grid points in one dimension
      PARAMETER (NGRID2=NGRID**2)      ! tot number grid points
      PARAMETER (NGRIDM1=NGRID-1)      ! number grid points minus 1
      PARAMETER (GSTEP=TOTSIZ/NGRID)   ! grid step size in n-miles

      PARAMETER (NBOAT=25)             ! number of boats
      PARAMETER (BSPEED=12.)           ! boat speed in knots
      PARAMETER (RADIUS=2.)            ! horizon dist. for observers in n-miles
      PARAMETER (GLADE=.9)             ! McGlade coefficient
      PARAMETER (BGMAX=1. , BGMIN=0.)  ! limits of goodness variable for boats
      PARAMETER (BKMIN=-3., BKMAX=8.)  ! associated limits of k

c      !--- I/O unit numbers---!
       integer INDAT,PLOTDAT,OBSDAT
       parameter (PLOTDAT=10, INDAT=11, OBSDAT=12)

       real sx(NSCHOOL),sy(NSCHOOL),dsx(NSCHOOL),dsy(NSCHOOL)
       real btx(NBOAT),bty(NBOAT),dbtx(NBOAT),dbty(NBOAT)
```

```
      real dxp(0:KLEVELS,0:100),dyp(0:KLEVELS,0:100)
      real qual(0:NQ,0:NQ)
      integer sfp(NSCHOOL),sbp(NSCHOOL),grid(0:NGRID-1,0:NGRID-1)
      integer iwaits(NSCHOOL),iwaitb(NBOAT)
      real good(NBOAT),qold(NSCHOOL)
      double precision dseed

      common/tops/ sx,sy,dsx,dsy,btx,bty,dbtx,dbty,
     >  istep,bstep,bstep2,rtest,sfp,sbp,grid,dseed,
     >  qual,qold,qfactor,good,
     >  dxp,dyp,iks1,iks2,ikb1,ikb2,sfact,bfact,
     >  dstmax,dstspan,scale,ndstepp,iqadd,iwaits,iwaitb
     ENDCLICHE


      block data
*******************************************************************************
*  preset some common variables
*******************************************************************************
      USE TOPSCOMM

      data grid/NGRID2*-1/,iwaits/NSCHOOL*0/,iwaitb/NBOAT*0/
      data dseed/1223435.d0/

      end


      PROGRAM TOPS
*******************************************************************************
*  mainline program module
*******************************************************************************
      USE TOPSCOMM

c     !---create dropfile---!
      call dropfile(0)

c     !---output files---!
      open(OBSDAT,file='obsdat',status='new',form='unformatted')
      open(PLOTDAT,file='plotdat',form='unformatted')
      write(PLOTDAT) NSCHOOL,NBOAT

c     !---assign positions & directions to vessels & schools---!
      call setup

c     !---speed up dolphins at start---!
      dstmax=dstmax*10.
      dstspan=dstspan*10.
      ndstepp=1

c     !---time loop start---!
      do 100 istep=0,MAXLOOPS
        if(istep.eq.50) then
c         !--- reduce dolphins to normal speed---!
          dstmax=dstmax/10.
          dstspan=dstspan/10.
          ndstepp=10
        endif
c       !---slide environmental quality pattern---!
        if(mod(istep,IQMOVE).eq.0) iqadd=mod(iqadd+1,NQ)
c       !---update school positions---!
        call updats
```

```
c       !---update vessel positions & record obs. data---!
        call updatb

 100   continue
c      !---time loop end---!

c      !---close up output file---!
       write(OBSDAT) -1 ; close(OBSDAT)
       write(PLOTDAT) -1. ; close(PLOTDAT)
       stop
       end


       subroutine setup
*************************************************************************
*  assign starting positions & directions to vessels & schools
*************************************************************************
       USE TOPSCOMM

       character name*12

c      !---dolfin school dist. per time step---!
       dstmax=DSPEED*STEP
       dstspan=.8*dstmax
       ndstepp=NDSTEP
c      !---boat dist. per time step---!
       bstep=BSPEED*STEP
       bstep2=2*bstep
       rtest=RADIUS**2-bstep**2

c      !---get direction tables---!
       open(INDAT,file='dtable',status='old',form='unformatted')
       read(INDAT) xkmin,xkmax
       read(INDAT) nk,deltak
       do 100 j=0,nk
         do 100 i=0,100
           read(INDAT) dxp(j,i),dyp(j,i)
 100   continue
       close(INDAT)
       iks1=(SKMIN-xkmin)*nk/(xkmax-xkmin)
       iks2=(SKMAX-xkmin)*nk/(xkmax-xkmin)
       iks1=max0(iks1,0)
       iks1=min0(iks1,nk)
       iks2=max0(iks2,iks1)
       iks2=min0(iks2,nk)
       sfact=(iks2-iks1)/(SGMAX-SGMIN)
       ikb1=(BKMIN-xkmin)*nk/(xkmax-xkmin)
       ikb2=(BKMAX-xkmin)*nk/(xkmax-xkmin)
       bfact=(ikb2-ikb1)/(BGMAX-BGMIN)

c      !---get env. quality topography---!
       qfactor=NQ/TOTSIZ
       anq=NPEAK*6.283185/NQ
       do 200 i=0,NQ
         qi=(amax1(0.,(1.+sin(i*anq-1.57079))/2.))**EX
         do 200 j=0,NQ
           qj=(amax1(0.,(1.+sin(j*anq-1.57079))/2.))**EX
           qual(i,j)=qi*qj
 200   continue
c      !--subroutine 'seed' seeds the random number generator 'randm'--!
       call seed(iseed)
```

```
c       !---set up schools---!
        do 300 is=1,NSCHOOL
c         !--'randm' returns uniformly dist. numbers from 0 to 1---!
          sx(is)=TOTSIZ*randm(dseed) ; sy(is)=TOTSIZ*randm(dseed)
          dir=6.28318*randm(dseed)
          dsx(is)=sin(dir) ; dsy(is)=cos(dir)
          write(PLOTDAT) sx(is),sy(is),iwaits(is)
c         !---get quality at prev. step so have first gradient---!
          dstep=dstmax-
     >          dstspan*qual(nint(sx(is)*qfactor),nint(sy(is)*qfactor))
          x = amod((sx(is)-dsx(is)*dstep*ndstepp+TOTSIZ),TOTSIZ)
          y = amod((sy(is)-dsy(is)*dstep*ndstepp+TOTSIZ),TOTSIZ)
          qold(is)=qual(nint(x*qfactor),nint(y*qfactor))
c         !---linked list entry---!
          ix=sx(is)/10 ; iy=sy(is)/10
          if(grid(ix,iy).eq.-1) then
            sfp(is)=-1
            sbp(is)=-1
          else
            sbp(is)=-1
            sfp(is)=grid(ix,iy)
            sbp(sfp(is))=is
          endif
          grid(ix,iy)=is
  300   continue
c       !---set up boats---!
        do 400 i=1,NBOAT
          dir=6.28318*randm(dseed)
          btx(i)=TOTSIZ*randm(dseed) ; bty(i)=TOTSIZ*randm(dseed)
          dbtx(i)=sin(dir) ; dbty(i)=cos(dir)
          write(PLOTDAT) btx(i),bty(i),dbtx(i),dbty(i),iwaitb(i)
  400   continue

        return
        end


        subroutine updats
*************************************************************************
*   update school positions,and repair list if necessary
*************************************************************************
        USE TOPSCOMM

c       !---indicate if dolphins choose new direction---!
        modistep=mod(istep,ndstepp)

        do 100 is=1,NSCHOOL
c         !---old box indices---!
          ix=sx(is)/GSTEP ; iy=sy(is)/GSTEP
c         !---get new direction if time to do so---!
          qnew=qual(nint(sx(is)*qfactor+iqadd),nint(sy(is)*qfactor))
          if(modistep.eq.0) then
            grad=(qnew-qold(is))/(qold(is)+QOLDMIN)
            xiks=iks1+(grad-SGMIN)*sfact
            qold(is)=qnew
            if(xiks.gt.iks2) then
              iks=iks2
            elseif(xiks.lt.iks1) then
              iks=iks1
            else
              iks=xiks
            endif
```

```
          iprob100= randm(dseed)*100
          dxnew=dyp(iks,iprob100)*dsx(is)+dxp(iks,iprob100)*dsy(is)
          dynew=dyp(iks,iprob100)*dsy(is)-dxp(iks,iprob100)*dsx(is)
          dstep=dxnew*dxnew+dynew*dynew
c         !---make sure direction vector stays unit length---!
          if(dstep.lt.0.99.or.dstep.gt.1.1) then
            dstep=sqrt(dstep)
            dsx(is)=dxnew/dstep ; dsy(is)=dynew/dstep
          else
            dsx(is)=dxnew ; dsy(is)=dynew
          endif
        endif
        dstep=dstmax-dstspan*qnew
c       !---move school, keeping w/in TOTSIZ mile Region---!
        sx(is)=amod(sx(is)+dsx(is)*dstep+TOTSIZ,TOTSIZ)
        sy(is)=amod(sy(is)+dsy(is)*dstep+TOTSIZ,TOTSIZ)
        call lmaint(ix,iy,is)
c       !---record school position---!
        write(PLOTDAT) sx(is),sy(is),iwaits(is)
        iwaits(is)=iwaits(is)-1
 100  continue
      return
      end


      subroutine lmaint(ix,iy,is)
*************************************************************************
*    repair school list pointers if necessary
*************************************************************************
      USE TOPSCOMM

c     !--new box indices---!
      ixn=sx(is)/GSTEP ; iyn=sy(is)/GSTEP
c     !---take care of move to new box---!
      if(ix.ne.ixn.or.iy.ne.iyn) then
c       !--extract from old list---!
        if(sbp(is).eq.-1) then
          grid(ix,iy)=sfp(is)  ! head of list
          if(sfp(is).gt.0) sbp(sfp(is))=-1
        else if (sfp(is).eq.-1) then
          sfp(sbp(is))=-1
        else
          sfp(sbp(is))=sfp(is)
          sbp(sfp(is))=sbp(is)
        endif
c       !---enter in new list---!
        if(grid(ixn,iyn).eq.-1) then
          sfp(is)=-1
          sbp(is)=-1
        else
          sbp(is)=-1
          sfp(is)=grid(ixn,iyn)
          sbp(sfp(is))=is
        endif
        grid(ixn,iyn)=is
      endif
      return
      end
```

```
      subroutine updatb
*************************************************************************
*   update boat positions
*  Output raw observer data...boat positions and schools sighted.
*************************************************************************
      USE TOPSCOMM

      logical within

      insght=0
      time=istep*STEP

c     !---loop thru vessels---!
      do 100 ib=1,NBOAT
c       !---remember current direction, needed regardless of boat status---!
        dx=dbtx(ib) ; dy=dbty(ib)
        isfirst=-1 ; yasfirst=1.e33
c       !---'iwaitb' > 0 means boat is processing a school---!
        if(iwaitb(ib).le.0) then     !--this boat is in searching mode--!
c         !---identify the first school in sight of vessel, if any, during this step---!
          x=btx(ib) ; y=bty(ib)
          i1=int(x/GSTEP)-2 ; i2=i1+5
          j1=int(y/GSTEP)-2 ; j2=j1+5
c         !---check for schools in sight---!
          do 20 i=i1,i2
c           !---deal with boat in edge strip---!
            if(i.lt.0) then
              im=1 ; ip=0 ; ii=i+NGRID
            elseif(i.gt.NGRIDM1) then
              im=0 ; ip=1 ; ii=i-NGRID
            else
              im=0 ; ip=0 ; ii=i
            endif
            do 20 j=j1,j2
              if(j.lt.0) then
                jm=1 ; jp=0 ; jj=j+NGRID
              elseif(j.gt.NGRIDM1) then
                jm=0 ; jp=1 ; ii=j-NGRID
              else
                jm=0 ; jp=0 ; jj=j
              endif
c             !---first school pointer for this square---!
              is=grid(ii,jj)
 10           if(is.gt.0) then  !- not end of list yet
                if(iwaits(is).le.0) then  !- is active school so check it
c                 !---check if within headstone-shaped observation area---!
                  within=.false.
                  delx=sx(is)+(ip-im)*TOTSIZ-x
                  dely=sy(is)+(jp-jm)*TOTSIZ-y
                  yas=dely*dy+delx*dx
c                 !---check if forward---!
                  if(yas.ge.0.) then
                    xas=abs(delx*dy-dely*dx)
c                   !---check if in range of boat track---!
                    if(xas.le.RADIUS) then
                      if(yas.le.bstep) then
c                       !--is in rectangular area swept--!
                        within=.true.
                      else
c                       !---check if in forward semicircle---!
                        if(yas*yas-yas*bstep2+xas*xas .le. rtest)
     >                                        within=.true.
```

```
                    endif
                  endif
                endif
                if(within) then     !- school within range
                  if(yas.lt.yasfirst) then  !- is nearest school so far
                    yasfirst=yas            !- so remember it
                    isfirst=is
                  endif
                endif
              endif      !- end of check for this school
c             !---go to next school in list---!
              is=sfp(is)
              go to 10
            endif
 20       continue   ! i-loop  ! j-loop
c         !---check if schools encountered and if so, take first one---!
          if(isfirst.gt.0) then    !--school was encountered
c           !---advance boat position to school position---!
            btx(ib)=sx(isfirst)
            bty(ib)=sy(isfirst)
c           !---set out-of-action signal for boat---!
            iwaitb(ib)=5
c           !---move school up to SPOOK miles away---!
            spk=randm(dseed)*SPOOK
            sx(isfirst)=
     >         amod(sx(isfirst)+dsx(isfirst)*spk+TOTSIZ,TOTSIZ)
            sy(isfirst)=
     >         amod(sy(isfirst)+dsy(isfirst)*spk+TOTSIZ,TOTSIZ)
c           !---set out-of-action signal for school---!
            iwaits(isfirst)=5

c           !---fix school pointers (boat pos. is school's old pos.)---!
            call lmaint(int(btx(ib)/GSTEP),int(bty(ib)/GSTEP),isfirst)
c           !---augment boat's goodness variable---!
            good(ib)=good(ib)+1.
          else        !--no school encountered
c           !---advance boat position one full step---!
            btx(ib)=amod(btx(ib)+dx*bstep+TOTSIZ,TOTSIZ)
            bty(ib)=amod(bty(ib)+dy*bstep+TOTSIZ,TOTSIZ)
          endif

c         !---take care of choice of direction for next time step---!

c         !---erode goodness variable---!
          good(ib)=GLADE*good(ib)
c         !---get k value---!
          ikb=ikb2-(good(ib)-BGMIN)*bfact
          if(ikb.gt.ikb2) then
            ikb=ikb2
          elseif(ikb.lt.ikb1) then
            ikb=ikb1
          endif
c         !---choose direction---!
          iprob100 = randm(dseed)*100
          dxnew=dyp(ikb,iprob100)*dbtx(ib)+dxp(ikb,iprob100)*dbty(ib)
          dynew=dyp(ikb,iprob100)*dbty(ib)-dxp(ikb,iprob100)*dbtx(ib)
          dbtx(ib)=dxnew
          dbty(ib)=dynew
        endif   !-- end of dealings for boat in searching status
```

```
c       !---record boat position, direction, & status---!
        write(PLOTDAT) btx(ib),bty(ib),dx,dy,iwaitb(ib)
        iwaitb(ib)=iwaitb(ib)-1
c       !---output observer data record---!
        if(isfirst.lt.0) then   ! isfirst<0 means no school sighted---!
          track=bstep
        else
          track=yasfirst
        endif
        write(OBSDAT) ib,track,btx(ib),bty(ib),isfirst

 100  continue  ! end of ib=1,NBOAT loop
      return
      end
```

Appendix B. Program for Circular-Normal distribution.

```
!************************************************************************
* Makes table of the inverse of the cumulative Circular-Normal
* probability function for various levels of the parameter, k.
* Directions expressed as unit vectors [dx, dy].
* For choosing new movement directions in the TOPS model.
*  Kleiber, 88/1/27
!************************************************************************

c       !---I/O parameters---
        integer TTY,INDAT,OUTDAT
        parameter (TTY=9)
        parameter (INDAT=10)
        parameter (OUTDAT=11)

       implicit real(a-z),integer(i,j,n)
       real p(0:1001),dxp(0:22,0:100),dyp(0:22,0:100)

       nk=22
       kmin=-3.
       kmax=8.
       deltak=(kmax-kmin)/nk
       factor=8*atan(1.)/1000.   ! 2*pi/1000

       do 100 ik=0,nk
c        !--choose k of the Circular-Normal probability function---
         k=kmin+ik*(kmax-kmin)/nk
c        !---get prob density for 1000 thetas---
         sum=0
         do 20 j=0,1000
              theta=j*factor
              p(j)=exp(k*cos(theta))
              sum=sum+p(j)
 20      continue
c        !---div. by sum and make cumulative---
         p(0)=p(0)/sum
         do 30 j=1,999
              p(j)=p(j-1)+p(j)/sum
 30      continue
c        !---pick theta for even 1/100ths of the probability space---
         p(1000)=1.
         j=0
         do 40 i=0,100
           pr=i/100.
           while (p(j).lt.pr)
              j=j+1
           repeat
           theta=j*factor
c          !---collect associated elements of unit vector w/ angle theta--
           dxp(ik,i)=sin(theta)    !-  (dx is sin(theta) because measure
           dyp(ik,i)=cos(theta)    !-     theta from straight ahead)
 40      continue
 100   continue  ! end ik=0,nk loop
```

```
      write(TTY,*) 'output to file starting'
      open(OUTDAT,file='dtable.dat',status='new',form='unformatted')
      write(OUTDAT) kmin,kmax
      write(OUTDAT) nk,deltak
      do 200 j=0,nk
        do 200 i=0,100
          write(OUTDAT) dxp(j,i),dyp(j,i)
200   continue

      end
```